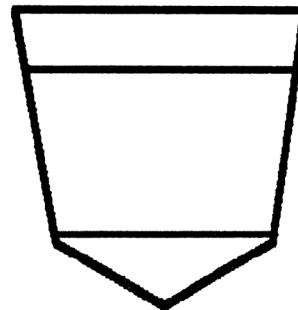


POCKET COMPUTER NEWSLETTER



© Copyright 1985 POCKET COMPUTER NEWSLETTER

Issue 38 - February

REVIEW OF TRAMSOFI TOOLKIT

Recently *PCW* had the opportunity to review the operation of a device designed to assist PC-1500 owners. The unit is named the TRAMSOFI Toolkit. It is currently being manufactured and distributed in Europe. This device proved interesting for several reasons, not the least of which is its ability to save and restore programs and/or data about 25 times faster than a standard PC-1500 CSAVE or CLOAD operation. What is more, this amazing increase in speed is accomplished through the use of a standard audio cassette recorder! (A Radio Shack model Minisetette-9 was used in tests conducted by *PCW*.)

The TRAMSOFI Toolkit consists of several electronic integrated chips and supporting components mounted inside a small shielded box, (measuring approximately 3 by 3 by 3/4 inches). A 60-pin male connector mounted on one end of the box is used to connect the unit directly to a PC-1500 or to the rear port of a CE-150 printer/cassette unit. There are two jacks on a side of the box, used to connect with an audio tape recorder.

One of the ICs inside the unit is a ROM containing software (apparently written in machine language) that controls overall operation of the unit. This software is used to "hook" onto the BASIC operating package in the PC-1500 and give it a number of new capabilities by providing new BASIC keywords.

For instance, there are four new keywords associated with using the enhanced magnetic tape storage system: FCHAIN, FLOAD, FSAVE and VERIFY. As you might surmise, FCHAIN permits you to chain in a new program segment using the faster tape technique, similar to what is done with a standard CHAIN statement. FLOAD and FSAVE invoke the faster tape system in the same fashion as CLOAD and CSAVE. These commands also provide the extensions that permit saving and loading machine language programs. The VERIFY command is similar to CLOAD?, but is intended verify data saved using the enhanced method. Furthermore, the enhanced tape system is able to verify machine language programs, something that is not possible using the standard CLOAD? option.

In tests conducted by *PCW*, using FSAVE to store a program on tape and FLOAD to recover it was many times faster than using CSAVE/CLOAD. For instance, a program that took about six minutes to load with Sharp's methodology, took barely 15 seconds with the TRAMSOFI Toolkit. All-in-all, the system seems to be about 25 times faster. This can make a big difference in how you approach using your PC. While six minutes seems interminable (and hence frequently not worth bothering with!), 15 seconds to load a decent-sized program into memory is relatively easy to bear.

The TRAMSOFI Toolkit would seem a pretty valuable device even if only considered for its enhanced program storage capability. But, the package contains a number of other features that will be of value to those who like to use their PC a lot. These other capabilities are referred to as

"programming aids." In essence, what this system enables you to do is "partition" user memory into modules, much as what occurs when you "merge" programs. The difference is that you have full, instantaneous control over the "modularization" and "merge" process.

Thus, you may have in effect, up to 255 "modules" in memory at a time. Only one module is "active" at any given moment. (The active module is the one in which any "editing" operations will take place.) This manipulation of modules is accomplished through the following types of new "keywords" that have been hooked into the BASIC operating system.

APPEND -- closes an active program module and enables you to create ("open") another module.

CHANGE -- is a powerful editing function that enables you to "search and replace" over a given range of line numbers or for a specified number of instances.

DELETE -- gives the capability to remove a range of line numbers from the current "active" module.

ERASE -- allows a user to eliminate an entire module.

FIND -- searches for a text string or BASIC token.

KEEP -- provides the ability to save the loadable part of cassette input following a loading error (partial load).

LINK -- combines program modules.

PLIST -- lists the current "active" program module.

PLAST -- gives last line number in the active program module.

PROGRAM -- activates a designated module.

RENUMBER -- provides capability to renumber the active module over any range and by any increment

SPLIT -- allows user to split one module into two modules (opposite of the LINK command).

It should be noted that this "modularization" is essentially an "editing" feature. That is, you use it when in the PROgram mode. When in the RUN mode, the PC operates in its normal fashion and must be directed to execute the desired "module" by reference to an appropriate label. Thus, the system does not provide true multiple file capability (such as is found in Hewlett-Packard's HP-71B). However, it does go a long way towards making it easier to develop complex programs or customize a PC by loading (and editing) essentially independent program modules.

Anyone seriously developing programs for the PC-1500 (and, probably, the Radio Shack PC-2), would undoubtedly appreciate this tool.

Alas, the TRAMSOFI Toolkit is not yet available in the United States. However, if sufficient genuine interest was expressed, it might be possible to have the tool imported and distributed. (Pricing in Europe is apparently equivalent to the price of a PC-1500.) If you would like to see this type of device made available in the U.S., *PCW* suggests you drop a note expressing your interest to: Chris Mallner, 150 Yantic Street - Apt. 159, Norwich, CT 06360.

FOR PC-1250/51/60/61 USERS

MACHINE CODE EXPOSED

Rick Wenger, 6221 - 18th Avenue, Kenosha, WI 53140, has provided the information contained in this article. Rick states that the work he has done would never have gotten off the ground without the pioneering efforts and encouragement of *Merlin Robey*.

He further advises that the information provided herein supersedes that published in Issue 26 of *PCN*. Note that the character code table and memory map for the PC-1250, contained in that earlier article, continue to be relevant with two corrections: the address range 0000 - 1FFF holds the so-called "hidden" ROM, and 4464 - 45DA contains bit maps used for character generation.

When using the following information, you should bear in mind that the PC-1250 appears to have a rather frail operating system from the point of view of machine language experimenters! Machine code should not be placed in REM statements or often-used string variables. It seems that even trying to print some "undefined" character codes can cause the PC to crash. Rick says that attempts to create a protected area for machine language code have not proved practical as M.L. programs cannot be called from the area above E000. He recommends that code be poked above any BASIC program and below dimensioned variables. (Addresses C400 or C500 seem good starting points.)

If a machine language routine is called from BASIC, the following CPU register should either be left undisturbed or restored to their initial values: SP, C1, 16, 58, 59, 5A and 5B. Other CPU registers appear to be fair game. (CPU register nomenclature is described later in this article.)

The "hidden" ROM has a number of interesting utilities. (Some of which, do not appear to be used in the 1250). For instance, the BCD values used in the calculation of various transcendental functions are stored beginning in the vicinity of address 08E8. The startup routine is at 0000.

Description of CPU Registers

Refer to the accompanying diagram for a pictorial view of the CPU registers referred to in this discussion.

There are ninety-six 8-bit registers (presumably these are located on the CPU chip) which Rick has numbered from 00 to 5F (using hexadecimal notation). These 96 registers are *not* part of the addressable 64K of memory. He refers to these 96 registers in the following presentation as the Central Processing Array (CPA). Some of these 96 registers serve special purposes as indicated in the accompanying table.

Besides the CPA registers, there are some other important CPU registers. These are shown in another table.

Note that there appears to be just two flags accessible to the machine language programmer. The Z (zero) flag is set or reset when certain types of operation result in a condition of zero or non-zero. The C (carry) flag is set or reset when operations such as an addition or subtraction requires or does not require a carry or borrow.

Rick feels that the existence of iterative BCD add and subtract instructions means that detection of half-carry or overflow status was not needed by the ROM programmers.

A list of the CPU instructions that Rick has discovered, along with brief explanations of the various types, is shown in an accompanying table. Instructions that do not contain a description are self-explanatory or analogous to others that are described in the listing.

This list should get the typical M.L. enthusiast off to a good start. (Remember, the same CPU chip is used in the Sharp PC-1250, PC-1251, PC-1260, PC-1261 and PC-1350 units as well as the Radio Shack PC-3.) Perhaps in a future issue of *PCN* we can reorganize this material by machine code order to assist those working on disassembly projects.

We think readers will agree it is a monumental piece of work. Congratulations, Rick!

Table The CPA Registers in the PC-1250 CPU.

CPA Register	Name	Description
00	C0	Primary counter. Used for iterative load instructions, etc.
01	C1	Another counter. Set to 01 throughout PC-1250 ROM to facilitate loading 2-byte words.
02	A	An 8-bit accumulator.
03	B	Combines with A to form a 16-bit register pair. B is also used to pass parameters to subroutines through the convenience of the EXA B instruction.
04	YL	Y-low, part of a 16-bit register pair capable of pointing to addressable memory.
05	YH	Y-high.
06	XL	X-low, part of a 16-bit register pair capable of pointing to addressable memory.
07	XH	X-high.
.	.	.
5A	(5A)	Stack storage. The stack works down from (5B). High byte is higher on the stack than low byte.
5B	(5B)	Start of subroutine stack.
5C	(5C)	Associated with low byte of data register D.
5D	(5D)	Associated with high byte of data register D.
.	.	.
5F	I	Top CPA register. Associated with OUT I instruction. Contents of this register used to control beeper, display, and "BUSY" annunciator.

Diagram PC-1250 CPU Registers.

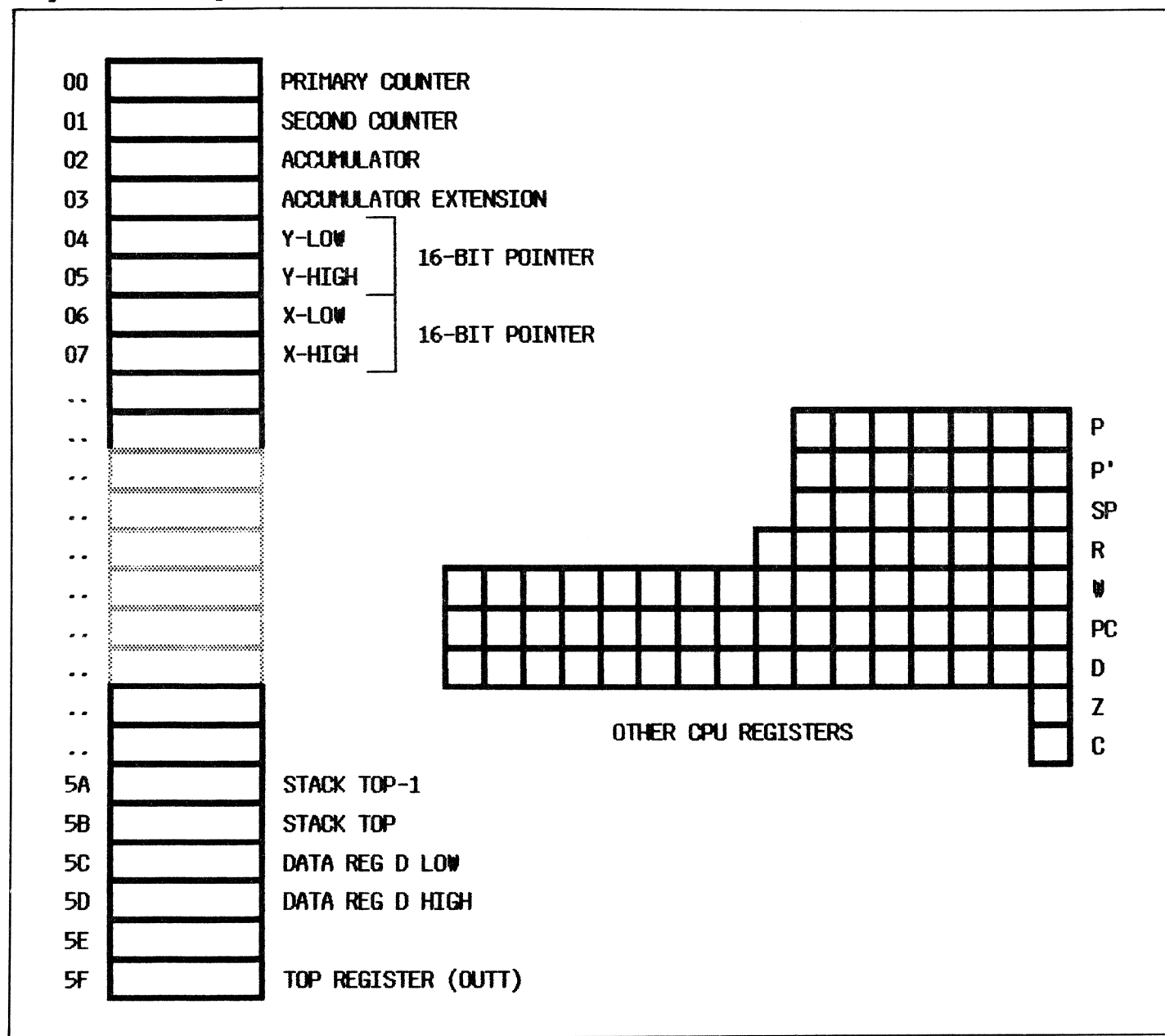


Table Other PC-1250 CPU Registers.

Register Size (Bits)	Name	Description
7	P	Primary pointer to the CPA.
7	P*	Secondary pointer to the CPA.
7	SP	Stack pointer to return address stack in CPA.
16	W	Points to addressable memory.
16	PC	Program counter. Indicates next instruction to be executed in memory.
16	D	Data register that obtains input from keyboard.
8	R	Working register. Retains last "operator" value.
1	Z	Zero flag.
1	C	Carry flag.

Table PC-1250 Machine Language Instruction Set.

LOAD INSTRUCTIONS

LDA *nn 02
Load accumulator (location 02 in the C.P.A.) with immediate data nn.

LDA P 20
Load accumulator with contents of P. P is not changed.

LDA P' 21

LDA SP 22

LDA (P) 59
Load accumulator with the contents of the C.P.A. register pointed to by P. P is not changed.

LDA (W) 57

LDA (PC) 56

LDB *nn 03

LDC0 *nn 00

LDC1 *nn 01

LDP *nn 12

LDP A 30

LDP' *nn 13

LDP' A 31

LDSP A 32

LDW *nnnn 10
Load 16-bit register W with next two immediate data bytes. High byte precedes low byte.

LDWL *nn 11
Low byte of W is loaded with immediate data nn. High byte of W is not changed.

LD (P)(W) 55
C.P.A. register pointed to by P is loaded with the contents of the memory location pointed to by W.

LD (P)(PC) 54

LD (W) A 52

LD (W)(P) 53

ADD INSTRUCTIONS

ADDA *nn 74
Immediate data nn is added to the accumulator. Carry and Zero flags are affected by the result.

ADD (P) A 44

ADD (P+) AB 14
A 16-bit addition operation. $P \gg P+1$. Carry and zero flags affected by the full 16-bit result.

ADC (P) A C4
Add with initial value of carry included in the calculation.

ADD (P) *nn 70

ADECO (P) A 0C
The BCD number in the location pointed to by P' is added to the BCD number pointed to by P. P and P' are then decremented. The process is repeated, (utilizing the Carry flag) the number of times specified by the contents of C.P.A. register C0. (P' is decremented an extra time for some reason.)

ADECO (P)(P') 0E

ITERATIVE LOADS

LDDA (Y) 25
Decrement 16-bit register Y, then load

the accumulator with the contents of the location pointed to by Y. New value of Y placed into W.

LDD (X) A 27

LDIA (Y) 24

LDI (X) A 26
Increment 16-bit register X. Then load the location pointed to by X with the contents of the accumulator. New value of X placed into W.

LD0 (P)(P') 08
Block C.P.A. transfer. Load the C.P.A. register pointed to by P with the contents of the C.P.A. register pointed to by P'. Repeat this process C0 times. Contents of C0 is not altered. The number of transfers done is equal to the contents of C0 plus 1.

LD0 (P)(W) 18
Block transfer from memory to C.P.A. registers. Identical to LD0 (P)(P') except W is not incremented after the last load (for some reason?).

LD0 (P) A 1E
The C.P.A. location pointed to by P is loaded with the contents of the accumulator. P is then incremented. The process is repeated C0 times.

LD0 (P) HR 35
Block transfer from hidden ROM to C.P.A. Starting address of hidden ROM must be in 16-bit C.P.A. register AB (with A, the accumulator, containing the low byte of the address value). C.P.A. registers pointed to by P through $(P+C0)$ are loaded. P goes to $P+C0+1$. (Hidden ROM in the PC-1250 has the address range 0 - 1FFF.

LD0 (W) A 1F

LD1 (P)(P') DA
Similar to LD0 (P)(P') except C1 is used as the counter.

LD1 (P)(W) 1A

EXCHANGE INSTRUCTIONS

EXA B DA

EXA (P) DB
Exchange contents of A and B. $[(P) \gg R, A \gg (P), R \gg A]$

ITERATIVE EXCHANGES

EX0 (P)(P') 09
Similar to LD0(P)(P') except contents are exchanged each go-around.

EX0 (P)(W) 19

EX1 (P)(P') 0B

EX1 (P)(W) 1B

SUBTRACT INSTRUCTIONS

SUBA *nn 75
Immediate data nn is subtracted from the accumulator. Carry and Zero flags are affected by the results.

SUB (P) A 45

SUB (P+) AB 15

SBC (P) A C5
Subtraction with initial value of the Carry flag included in the calculation.

SUB (P) *nn 71

SDECO (P) A 0D

SDECO (P)(P') 0F
BCD number at location pointed to by P' is subtracted from BCD value pointed to by P. P and P' decremented. Process repeated number of times indicated by contents of C0. The Carry and Zero flags are affected by the result of the total answer, not just the highest or lowest byte.

SHORT FORM LOADS

SLP 00 80
Short-form Load P with xx instruction. Single-byte instruction. No immediate data byte required.

SLP 01 81

...

...

...

JUMP INSTRUCTIONS

JMP nnnn 79
Jump unconditionally to nnnn.

JCC nnnn 7D
Jump if Carry flag is cleared to address nnnn.

JCS nnnn 7F
Jump if Carry flag is set to address nnnn.

JZC nnnn 7C
Jump if Zero flag is cleared to address nnnn.

JZS nnnn 7E
Jump if Zero flag is set to address nnnn.

FORWARD BRANCHES

Note: When a forward or reverse branch is performed, the low byte of the new address for the PC is computed and stored in (SP-1).

FWD nn 2C
Jump forward unconditionally to current address + nn.

FCC nn 2A
Jump forward if Carry cleared to current address + nn.

FCS nn 3A
Jump forward if Carry Set to current address + nn.

FZC nn 28
Jump forward if Zero cleared to current address + nn.

FZS nn 38
Jump forward if Zero set to current address + nn.

REVERSE BRANCHES

REV nn 2D
Jump reverse unconditionally to current address - nn.

RCC nn 2B

RCS nn 3B

RZC nn 29

RZS nn 39

RDR nn 2F
Reverse branch with decrement and repeat. (SP) is decremented by one and a reverse branch is done if Carry is not set (cleared). If Carry is set, SP is incremented and execution continues sequentially. (One would normally push the number of repetitions desired onto the stack before using an RDR loop. Thus, it is unwise to call a subroutine from within an RDR loop.)

ROTATE INSTRUCTIONS

RDA 58
Rotate digits in accumulator. (Exchange high and low nibbles.)

RLA 5A
Rotate bits in accumulator to the left through the Carry flag.

RRA D2
Rotate bits in accumulator to the right through the Carry flag.

RLO (P) ID
Rotates nibbles in a portion of the C.P.A. (registers P-C0 through P) to the left. Right-most nibble becomes zero. Left-most nibble is lost.
P-C0-1>>P.
Rotates nibbles in a portion of the C.P.A. to the right. P+C0+1>>P.

RRO (P) 1C

INCREMENT INSTRUCTIONS

INCA 42

INCB C2

INCC0 40

INCCI 60

INCP 50

P is incremented by one. Carry and Zero flags are not affected.

INCX 06

The 16-bit X register pair is incremented. The Carry and Zero flags are not affected. New X>>W.

INCY 04

INC (08) 48

The C.P.A. location pointed to by the contents of C.P.A. register 08 is incremented by one. The Carry and Zero flags are affected.

INC (09) 68

INC (0A) 4A

INC (0B) 6A

DECREMENT INSTRUCTIONS

DEA 43

DEB 63

DECO 41

DECI C1

DEP 51

DEX 07

DEY 05

DE (08) 49

DE (09) C9

DE (0A) 4B

The C.P.A. location pointed to by the contents of C.P.A. register 0A is decremented by one.

DE (0B) CB

BOOLEAN OPS & COMPARES

ANDA *nn 64

Contents of accumulator are ANDed with immediate data nn.

ORA *nn 65

BITA *nn 66

CPA *nn 67

AND (P) A 46

OR (P) A 47

BIT (P) A C6

CP (P) A C7

Flags C and Z Set/Reset based on (P)-A, although subtraction operation is virtual and does not change register contents.

AND (P) *nn 60

OR (P) *nn 61

Contents of C.P.A. location pointed to by P are Ored with immediate data nn.

BIT (P) *nn 62

CP (P) *nn 63

AND (W) *nn D4

OR (W) *nn D5

BIT (W) *nn D6

The Zero flag is set or reset based on the result obtained when the memory location pointed to by W is ANDed with immediate data byte nn. The Carry flag and other registers are not affected by the operation. (SP-1) used as scratchpad for Boolean ops involving W register.

ZBITA *nn 76

The Zero flag is set or reset based on results of A ANDed with nn, except that if Z is reset at the start, it remains reset. The Carry flag is not affected. R>>C0.

ZCPA *nn 77

ZBIT (P) *nn 72

ZCP (P) *nn 73

The Zero and Carry flags are set or reset based on results of (P) - nn, except that if Z is cleared at the start, it remains cleared. R>>C0.

SUBROUTINE INSTRUCTIONS

JSR nnnn 78

Jump to subroutine at address nnnn.

RTS 37

Return from subroutine.

SHORT FORM CALLS

SS 00 nn E0

Short-form jump to subroutine within hidden ROM. A two-byte call instruction.

SS 01 nn E1

.... ..

.... ..

.... ..

SS 1F nn FF

MISCELLANEOUS

SET C D0

Z flag is set

CLR C D1

Z flag is set

REFR *nn 4E

Suspends CPU operation for a length of time proportional to immediate data byte nn. Reroutes current to lines opened by OUTT.

IN nn 6B

Sets or resets Zero flag based on the status (active or non-active) of the RSV/PRO/RUN/OFF slide switch or BRK key. Data byte nn determines the signal that is to be tested.

OUT T DF

OUT T SUMMARY

Bit 0 - Display on if 1, Busy on if 0

Bit 3 - Power down if 1

Bit 4 - Tighten Beeper if 1

Slacken Beeper if 0

Bit 5 - Beeper on if 1

Bit 6 - Distorts Beeper if 1

Bits 1,2,7 - ????

INC P IF C0 6F

PUSH A 34

POP A 5B

PUSH XX *nnnn 7A

Pushes the address value nnnn onto the stack. First nn byte is higher in the stack. Do not know purpose of the XX byte in this 4-byte instruction.

CLR (SP) D8

CLA 23

INA D 4C

NOP 33

FOR HEWLETT-PACKARD HP-71B USERS

LEARN MORSE CODE

Here is an interesting program that you can use to learn (or refresh) Morse code skills. It is capable of generating these signals over a range of speeds suitable for training purposes. The audible tone it produces can be selected to suit an individual user.

The code is sent as cipher groups. There are five characters per group. This method is highly effective as a training technique because the student cannot guess or deduce what character will be sent next, as might be the case if ordinary text was utilized.

As the audio Morse code sound is produced by the program, the corresponding character is shown on the display. Thus, a novice can use the program to "learn" the code just by listening and watching the LCD screen. Once some proficiency has been developed, the student can start writing down the characters without looking at the display. The program stores the cipher code it sends in a buffer. At the end of a block of 200 characters, the user can opt to have the cipher

groups shown on the display for confirmation purposes.

Once the program has been loaded into memory (the listing shown has a byte count of 1343), it is started with a simple RUN command. Tone and speed values should be given as an integer in the range 1 to 5. (Tone and speed increase as the associated value increases.) Once the speed value has been inputted, operation is automatic until a group of 200 characters has been sounded. To view the characters that have been sent at the conclusion of a block, respond with a Y to the query WANT LISTING?

Close examination of lines 200 - 280 in the listing should yield an understanding of how characters are generated by the program. You can add or delete characters by placing appropriate data in the A\$ array. (Don't forget to change the DIM statement in line 10 and the RND statement in line 70, if additions or deletions are made.)

If you have a program for the HP-71B that you would like to share with *PCW* readers, or tips for HP-71B users, drop a note to the editor. The -71B is a powerful hand-held that deserves to receive support and recognition!

Program Morse Code on the HP-71B.

```
10 DELAY 0 @ DESTROY ALL @ OPTION BASE 0 @ DIM A$(39)[8],B(5),C(5),C$(199)[1] @ GOSUB 200
20 INPUT "TONE? ";T @ T=ABS(INT(T)) @ IF T<1 OR T>5 THEN BEEP @ GOTO 20
30 T=T*200 @ T=T+400
40 INPUT "SPEED? ";S @ S=ABS(INT(S)) @ IF S<1 OR S>5 THEN BEEP @ GOTO 40
50 S=S*.02 @ S=.12-S @ P=S
60 DISP "" @ L=0 @ RANDOMIZE
70 R=INT(RND*40)
80 C$(L)=A$(R)[1,1] @ DISP TAB(11);C$(L)
90 D=VAL(A$(R)[2,2])
100 FOR J=3 TO D+2 @ C(J-3)=VAL(A$(R)[J,J]) @ NEXT J
110 FOR J=0 TO D-1
120 IF C(J)=1 THEN 140
130 BEEP T,S @ GOTO 150
140 BEEP T,S*3
150 NEXT J @ WAIT S @ L=L+1 @ IF L/5=INT(L/5) THEN WAIT 5*S
160 IF L#200 THEN 70
170 INPUT "WANT LISTING? ";W$ @ IF UPRC$(W$[1,1])="Y" THEN GOTO 300
180 GOTO 10
200 A$(0)="A201" @ A$(1)="B41000" @ A$(2)="C41010" @ A$(3)="D3100" @ A$(4)="E10"
210 A$(5)="F40010" @ A$(6)="G3110" @ A$(7)="H40000" @ A$(8)="I200" @ A$(9)="J40111"
220 A$(10)="K3101" @ A$(11)="L40100" @ A$(12)="M211" @ A$(13)="N210" @ A$(14)="O3111"
230 A$(15)="P40110" @ A$(16)="Q41101" @ A$(17)="R3010" @ A$(18)="S3000" @ A$(19)="T11"
240 A$(20)="U3001" @ A$(21)="V40001" @ A$(22)="W3011" @ A$(23)="X41001"
250 A$(24)="Y41011" @ A$(25)="Z41100" @ A$(26)="1501111" @ A$(27)="2500111"
260 A$(28)="3500011" @ A$(29)="4500001" @ A$(30)="5500000" @ A$(31)="6510000"
270 A$(32)="7511000" @ A$(33)="8511100" @ A$(34)="9511110" @ A$(35)="0511111"
280 A$(36)="6010101" @ A$(37)="6110011" @ A$(38)="76001100" @ A$(39)="/510010"
290 RETURN
300 DELAY 8 @ STD
310 FOR I=0 TO L-1
320 W$=STR$(I+1) @ IF I<9 THEN W$="00"&W$ @ GOTO 340
330 IF I<99 THEN W$="0"&W$
340 IF I=0 THEN 360
350 IF I/10#INT(I/10) THEN 370
360 DISP W$;" ";
370 DISP C$(I); @ IF (I+1)/5=INT((I+1)/5) THEN DISP " ";
380 IF (I+1)/10=INT((I+1)/10) THEN DISP ""
390 NEXT I
400 GOTO 10
```

FOR PC-1500 & PC-2 USERS

PASSWORD PROTECTION

Last issue, this column discussed some of the ramifications of attempting to protect programs from unauthorized use or duplication. As indicated in that article, such protection is generally "breakable" by those that have knowledge of the system being utilized. However, they can be effective against the "casual" user or those that do not make a concerted effort to defeat the mechanism(s).

The program described here, submitted by *Eric Bowman, PEA 81, Exeter, NH 03833*, is claimed to be a "fairly effective password protection system." The protection program is also capable of disabling use of the printer by those not authorized. The program has been designed (as presented here) to reside in the machine language program area of a PC-1500A. However, the necessary changes for relocation of the program, so that it might be used in a standard PC-1500 (or Radio Shack PC-2) have been included in the article.

Operating instructions are as follows. After typing in the program, immediately execute a CALL 67D33. The computer will then prompt the user with "SET PASSWORD:". At this point, the user should decide on a password of five letters and type it in. As the fifth letter is entered, the password will be displayed, along with a question mark. If the password is the desired one, "Y" should be pressed to install the password and terminate the program. If the displayed

password is not appropriate, responding with an "N" will restart the program, thus enabling the user to try again. Pressing BREAK at any time will terminate the program leaving the previous password in effect.

To actually use the password, execute a CALL 67C01. At this point the computer should turn off by itself. Upon power-up, the display will read: PC-1500A ONLINE:. Pressing any key, including BREAK, should cause PASSWORD: to be displayed. At this point, the user must enter the previously defined keyword. As the fifth letter of the password is entered, the computer will either turn itself off, indicating an invalid password, or it will display the query PRINT (Y/N)? The latter indicates that the user has access to the PC and now has the option of enabling or disabling the printer. If the user answers "Y" at this time, the printer will be available for normal use. Responding with a "N" will result in the PC simulating the low-battery condition if any attempt is made at using printer commands. (This yields an ERROR 78 message.) This state may be altered by either running the program again and answering "Y" to the printer prompt or by a POKE 679F1,0 directive.

(Note: during the password entry process, the LF key is disabled.)

Program Registers

The program uses various memory locations throughout memory

Program Password Protection.

7C01 B5 40 AE 26	7C71 FD 88 B5 36	7CE1 50 58 78 5A	7D51 52 44 3A BE
7C05 4E B5 43 AE	7C75 AE 78 75 BE	7CE5 60 6A 0A F5	7D55 ED 3B B5 4E
7C09 76 4F EB 79	7C79 E2 43 83 5A	7CE9 88 03 F2 BA	7D59 AE 78 75 BE
7C0D F1 FF FD BE	7C7D FD C8 B5 2A	7CED 7C 01 F2 FD	7D5D E2 43 C3 42
7C11 E9 79 D5 00	7C81 BE ED 4D FD	7CF1 58 B5 0B FD	7D61 FD C8 BE ED
7C15 FD 58 B5 0A	7C85 8A FD 0A 41	7CF5 CA FD 6A 4A	7D65 4D FD 8A FD
7C19 FD CA FD 6A	7C89 A5 79 D5 DD	7CF9 0C BA 7D 09	7D69 0A 41 4E 05
7C1D 4A 10 8E 10	7C8D B7 05 8B 07	7CFD 50 52 49 4E	7D6D 8B 04 FD 88
7C21 50 43 2D 31	7C91 AE 79 D5 FD	7D01 54 20 28 59	7D71 9E 17 F2 68
7C25 35 30 30 41	7C95 88 9E 20 35	7D05 2F 4E 29 3F	7D75 7F 6A 00 4A
7C29 20 4F 4E 4C	7C99 D0 AE 7A 04	7D09 BE ED 3B BE	7D79 05 BE ED 3B
7C2D 49 4E 45 3A	7C9D B5 76 AE 7A	7D0D E2 43 83 10	7D7D B5 24 AE 78
7C31 BE ED 3B 48	7CA1 05 B5 50 AE	7D11 B7 59 8B 13	7D81 75 B5 3F BE
7C35 78 4A 60 58	7CA5 7A 06 B5 05	7D15 B5 FF AE 79	7D85 ED 57 BE E2
7C39 78 5A 50 6A	7CA9 AE 7A 07 B5	7D19 F1 E9 79 FF	7D89 43 C3 42 B7
7C3D 0A F5 88 03	7CAD D0 AE 7A 14	7D1D 00 FD 81 9A	7D8D 59 8B 07 B7
7C41 B5 00 AE 79	7CB1 B5 79 AE 7A	7D21 FD E9 F0 0B	7D91 4E 8B 11 BA
7C45 FF 48 6F 4A	7CB5 15 B5 FA AE	7D25 FD 9E 39 E9	7D95 7D 5C 48 7F
7C49 FF CA 65 CA	7CB9 7A 16 B5 05	7D29 79 F1 00 E9	7D99 4A 00 58 79
7C4D 67 CA 69 BE	7CBD AE 7A 17 B5	7D2D 79 FF 00 FD	7D9D 5A FA 6A 04
7C51 E3 3F F2 FD	7CC1 04 BE D0 F9	7D31 81 9A 48 7F	7DA1 F5 88 03 9A
7C55 58 B5 0B FD	7CC5 8B 17 48 78	7D35 4A 00 FD 88	7DA5 F2 48 7F 4A
7C59 CA FD 6A 4A	7CC9 4A 50 58 78	7D39 F2 FD 58 B5	7DA9 00 FD 88 BA
7C5D 09 BA 7C 6A	7CCD 5A 60 6A 0A	7D3D 0B FD CA FD	7DAD 7D 39 CC 65
7C61 50 41 53 53	7CD1 F5 88 03 BA	7D41 6A 4A 0D BA	7DB1 CA 69 49 00
7C65 57 4F 52 44	7CD5 7C EF FD E9	7D45 7D 54 53 45	7DB5 45 DD 99 04
7C69 3A BE ED 3B	7CD9 F0 0B FD 9E	7D49 54 20 50 41	7DB9 46 CA 67 9A
7C6D 48 76 4A 50	7CDD 66 48 78 4A	7D4D 53 53 57 4F	

to store parameters. Most of these locations are, it is believed, used by the CE-158 interface. Thus, it is not advisable to attempt using this program when a CE-158 is connected without first making sure that the following locations will not be disturbed:

79D5 Counts the number of characters entered
7650-4 Stores the characters being typed
79FA-E Location of actual password
7F00-4 Stores the characters being typed in the SET PASSWORD routine

Several characteristics of the program are worth mentioning. Once access has been gained, the computer is set to DEG, RUN, softkey mode I, and LOCK. If this is not desirable, everything except the setting of LOCK may be changed by replacing the first ten bytes of the program with NOPs. Note too, that the password is not in effect unless the machine is turned off by using CALL 67C01. (The OFF key functions normally.) You might want to place the CALL 67C01 directive under control of a softkey. Also, if the machine sits idle for more than seven minutes during the password entry process, it will automatically shut off. However, upon power-up, nothing will have changed and the user can continue entering a password.

Relocating the Program

To simplify the process, it is suggested that you relocate the program so that it starts at an address ending with 01. Thus, if you elected to start it at address 3901, the following changes would need to be made:

385F to 39
39D5 to 39
39ED to 39

Available Only by Prepaid Subscription for a Calendar Year Period (January - December). You are sent back issues for the calendar year to which you subscribe, at the time you enroll.

- Enroll me as a 1985 Subscriber (Issue numbers 37-44). \$24.00 in U.S. (U.S. \$30.00 to Canada/Mexico. Elsewhere U.S. \$40.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1984-85 Subscriber (Issue numbers 31-44). \$42.00 in U.S. (U.S. \$51.00 to Canada/Mexico. Elsewhere U.S. \$70.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1983-85 Subscriber (Issue numbers 21-44). \$78.00 in U.S. (U.S. \$93.00 to Canada/Mexico. Elsewhere U.S. \$120.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1982-85 Subscriber (Issue numbers 11-44). \$102.00 in U.S. (\$125.00 to Canada/Mexico. Elsewhere U.S. \$160.00 payable in U.S. funds against a U.S. bank.)
- Check here if paying by MasterCard or VISA. Please give credit card information below.

Orders must be accompanied by payment in full.
All checks must be magnetically encoded, payable in U.S. funds and drawn against a U.S. bank.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA #: _____

Signature: _____ Exp. Date: _____

mail this order form to:

POCKET COMPUTER NEWSLETTER

P.O. Box 232, Seymour, CT 06483

39FB to 3A
3A45 to 3A
3A95 to 3A
3AAD to 3A

The program uses 67F00-4 as a brief stack. If the user does not have a PC-1500A (or has another program stored there), this stack can be relocated to 7650-4 by making the following alterations:

7D34 (3A34) to 76
7D36 (3A36) to 50
7D75 (3A75) to 76
7D77 (3A77) to 50
7D98 (3A98) to 76
7D9A (3A9A) to 50

Of course, by storing the appropriate high and low bytes at these locations, the stack could be relocated anywhere in memory. However, 7650 seems a good place.

Memory Map

The main sections of the program are located as follows:

7C01 Start of password routine. This section turns the computer off, then asks for the password upon power-up.
7D33 Start of routine to set the password. Asks for the password, verifies it and stores it.
7DAF "Renew" program originally contributed by Norlin Rober (PCW Issue 36, page 16).
79FA-FE Location of password.

In the event that the password is ever forgotten, you can perform the following procedure. First, turn the machine on while pressing the "all reset" button (on the back of the PC) and holding the ON key. Press the clear key and then enter and execute the following brief machine language program. (It may be located at any convenient place in memory).

48 78 4A 50 58 78 5A 60 6A 0AF 5 88 03 9A

This routine will set the BASIC parameters back to their original values and will allow you to re-use whatever BASIC program was in memory.

So there you have it! Why not give it a try. How difficult do you think it would be to break this protection scheme (assuming you didn't have the information given in this article)?

FOR PC-1350 USERS

TITLE

If you think this column is a little short this issue, go back and take a look at the PC-1250 column. See what it says there? The CPU used in the Sharp PC-1350 is the same one used in the 1250! That means the instruction set that has been so nicely decoded by Rick Wenger (and presented in this issue) for the 1250, also works on the 1350!

Want to verify that is indeed the case? Try loading the following short machine language routine into your PC-1350:

POKE 66800,612,65F,2,620,60B,60F,637

Then, place the following BASIC statement in memory:

100 "B" CALL 66800 : WAIT 1 : PRINT "*" : GOTO "B"

Executing the BASIC routine using DEF/B should result in your hearing a string of "beeps" coming from your PC and a series of asterisks appearing on the display. The beeps are being generated by the machine language routine.

Can you decode the five machine language instructions that make up this routine (using the information provided in this issue under the 1250 section)? Hint: the first one is LDP # 5F. Once you do this, you can try some experimenting on your own.

Hopefully, we will have some PC-1350 memory maps soon!